

Linguagem XML

(“eXtensible Markup Language”)

Baseado nos slides do professor Paulo Trigo
Todas as alterações são da responsabilidade do professor António Teófilo
e do professor Diogo Remédios

Alguns passos e marcos em torno da linguagem XML

- Evoluiu da linguagem SGML (“Standard Generalized Markup Language”)
 - ISO 8879 em 1986, <http://www.w3.org/TR/NOTE-sgml-xml-971215>
 - SGML baseia-se em marcas (“tags”) para definir semântica e a sintaxe de texto.
- A concretização mais conhecida da linguagem SGML é o
 - HTML (“Hyper Text Markup Language”), definido no RFC 1866, em 1995 (1991)
 - ... como um conjunto finito de marcas com método universal de apresentação
- A linguagem XML 1.0 surge em 1998 como versão simplificada de SGML
 - <http://www.w3.org/TR/2006/REC-xml-20060816/>
 - ... espaço de nomes (“*namespaces*”) para mesmas marcas em várias aplicações
 - XML 1.1 (2004) adiciona a capacidade de utilizar mais caracteres nos identificadores dos elementos e nos atributos
 - Versões correntes: XML 1.0 versão 4, XML 1.1 versão 2
- Apresentação de XML recorre à “eXtensible Stylesheet Language” (XSL)
 - XSLT (para transformação de documentos XML)
 - XSL-FO (para detalhes da apresentação, e.g. “browser”, “postscript”...)
- A linguagem XHTML é uma reformulação do HTML 4 baseada no XML 1.0
 - reforça HTML com a validação adoptada no XML (<http://www.w3.org/TR/xhtml1/>)

... linguagem baseada em marcas (“tags”)

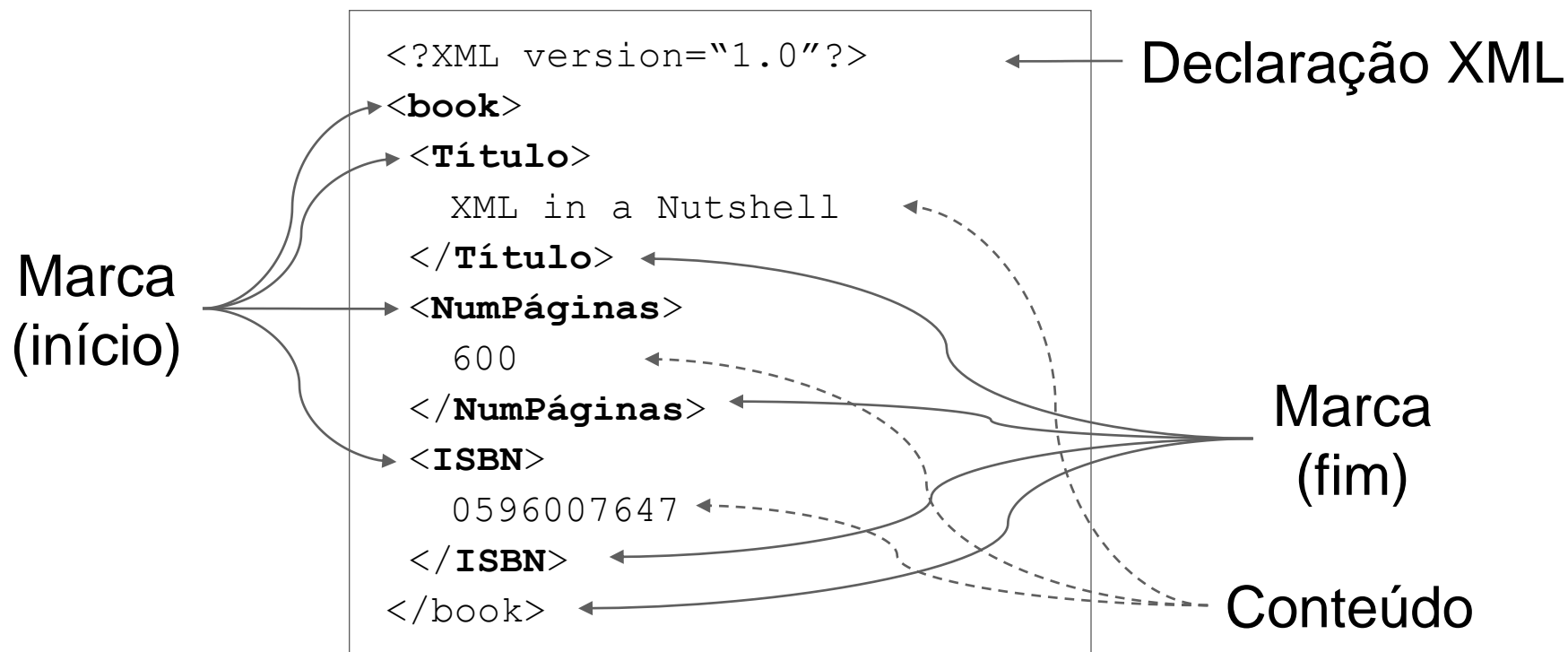
- Uma linguagem baseada em marcas deve especificar
 - as marcas que são permitidas
 - as marcas que são obrigatórias
 - como é que as marcas se distinguem do texto
 - qual a estrutura de marcas que deve ser seguida
- A linguagem XML permite especificar marcas
 - não impõe uma estrutura às marcas que se utilizam
 - ... mas obriga a respeitar as regras de formação de marcas
- A linguagem HTML assume marcas predefinidas
 - e as aplicações tratam essas marcas de forma predefinida
 - ... o respeito pelas regras é por vezes relaxado pelas aplicações
 - ... e.g. o browser pode admitir erros na formação das marcas

Exemplo de texto e sua estrutura em XML

Texto com dados separados por vírgula (sequência de caracteres):

```
"XML in a Nutshell", "600", "0596007647"
```

O mesmo texto com os dados estruturados em XML:



De slide para ficheiro: nesta passagem as plicas " e " devem ser passadas para "

HTML e XML

- O HTML foi desenvolvido para especificar hiper-texto*
 - ❑ assume um conjunto predefinido de marcas
 - ❑ não é vocacionado para construir outras linguagens de marcas
 - ❑ cada marca representa também informação de apresentação
 - embora possa ser redefinida separadamente (e.g. via CSS)
- O XML tem o objectivo de especificar linguagens de marcas
 - ❑ uma linguagem de marcas especifica-se para determinado domínio
 - ❑ ... no entanto o XML não está comprometido com qualquer domínio!
- ... marcas formam a estrutura de um qualquer documento
 - ❑ e não a forma como esse documento deve ser apresentado
- ... apresentação recorre ao XSL (“eXtensible Stylesheet Language”)
 - ❑ XSLT (“XSL Transformations”),
 - ❑ XPath, e
 - ❑ XSL-FO (“XSL Formatting Objects”).

* Texto com referências e anotações

Uma receita em HTML (“HyperText Markup Language”)

```
<html>
  <head><title>Receita</title><link rel="stylesheet" href="apresentacao.css"/></head>
  <body>
    <h1>A receita, de nome original "Sumeshi",<br>é usualmente designada "Arroz de Sushi".</h1>
    <p class="fundoCorA">O êxito do Sushi depende de um bom Sumeshi.</p>
    Ingredientes:
    <table border="1">
      <tr><td>600 g</td><td>Arroz de grão curto (nishiki)</td></tr>
      <tr><td>750 ml</td><td>Água</td></tr>
      <tr><td>125 ml</td><td>Vinagre de arroz</td></tr>
      <tr><td>55 g</td><td>Açúcar</td></tr>
      <tr><td>0.5 colher de chá</td><td>Sal</td></tr>
    </table>
    <p class="fundoCorB">O trabalho de preparação segue um processo que dura cerca de 1h15.</p>
    Passos de preparação:
    <table border="1">
      <tr><td>Lavar o arroz até que a água saia limpa e
        repousar num coador pelo menos 30 minutos.</td></tr>
      <tr><td>Os restantes 45 minutos são completo segredo!</td></tr>
    </table>
  </body>
</html>
```

receita.html

... HTML & apresentação definidas separadamente

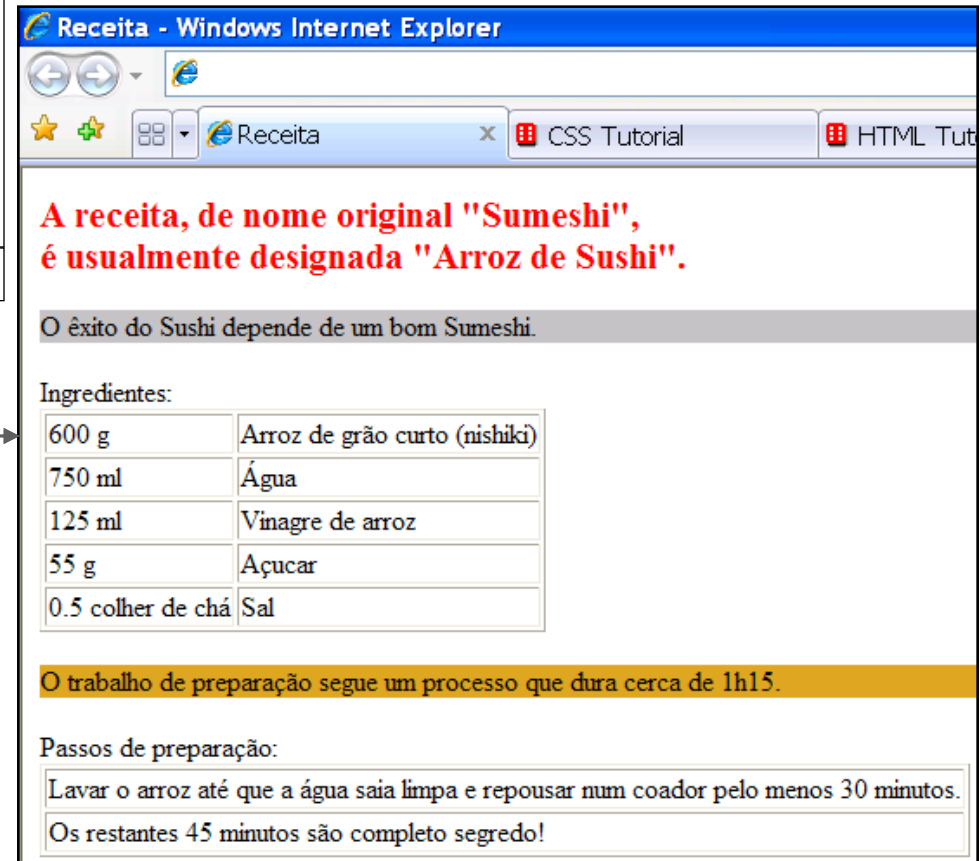
```
<html>
  <head>
    <title>Receita</title>
    <link rel="Stylesheet" href="apresentacao.css"/>
  </head>
  ... ver folha anterior
```

receita.html

CSS ≡ “Cascading Style Sheets”

```
/*
Estrutura geral de um comando CSS
selector {property: value}
*/
p.fundoCorA {background-color:#C0C0C0}
p.fundoCorB {background-color:GoldenRod}
h1 {color: red; text-align: left; font-size: 16pt }
```

apresentacao.css

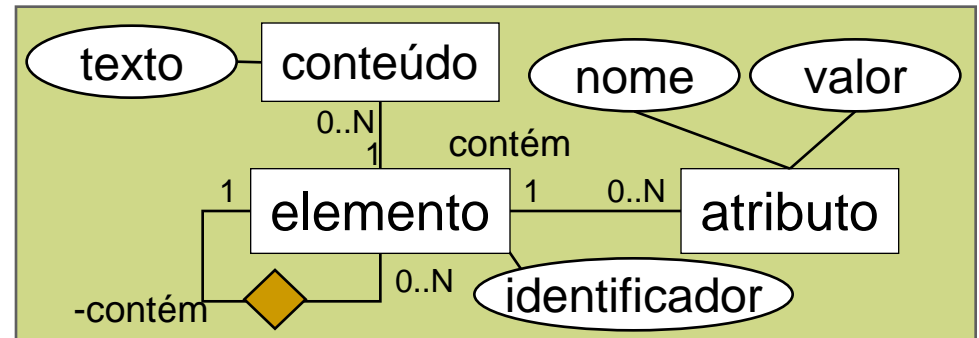


O que o XML não é?

- Não é uma linguagem de programação
 - ❑ o XML não se compila (nem interpreta) para gerar código executável
 - ❑ o XML descreve formatos; os programas usam-nos para **fazer** coisas, tal como se lê um ficheiro de configuração e se faz o que lá está descrito
 - ❑ o XML simplesmente **é**; o XML não **faz**!
- Não é um protocolo de transferência de dados
 - ❑ o XML não envia dados (nem o HTML); quem envia é o HTTP, FTP, ...
 - ❑ os dados enviados podem ter a estrutura XML (ou HTML)
- Não é uma base de dados (BD) nem um sistema de gestão de BD
 - ❑ o XML não substitui o MySQL, SQLServer, etc
 - ❑ uma BD pode conter dados XML, mas a BD não é um documento XML!
 - ❑ o XML pode usar-se como formato para transferir dados entre as aplicações e os SGBDs; no entanto, continua a ser apenas um formato!

Estrutura de um documento XML

- Um documento XML é constituído:
 - Um único elemento raiz
- Um Elemento pode conter:
 - Atributos
 - outros Elementos
 - conteúdo textual (verde)
 - Uma tag de início e uma de fim
 - Tags de início em Azul
 - Com Identificador com rodeado por '<' e '>'
 - uma tag de fim tem o mesmo id
 - e é rodeada por '</' e '>'
 - Uma única tag rodeada por '<' e '>'
 - Eventualmente com atributos
 - Nos casos sem elementos nem texto
- Um Atributo:
 - tem um nome e um valor
 - Com o nome em Vermelho
 - Existem dentro da tag de início ou tag única
- Sintaxe:
 - <idElem atrib1="valor" atrib2="valor" ...>conteudos e outros Elementos</idElem>
 - <idElem atrib1="valor" atrib2="valor" .../>



este modelo descreve 80% do XML!

Exemplo

```
<books>
  <book num="010">
    <title>The Da Vinci code</title>
    <isbn>978-0385504201</isbn>
  </book>
  <book num="011">
    <title>Angels and Demons: A Novel</title>
    <isbn>978-0743493468</isbn>
  </book>
  <book num="012">
    mais texto
    <title>Digital Fortress: A Thriller</title>
    mais texto
    <isbn>978-0312944926</isbn>
    mais texto
  </book>
</books>
```

... a mesma receita escrita em XML

Estrutura Global:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita>
  <nomeOriginal>Sumeshi</nomeOriginal>
  <nomeUsual>Arroz de Sushi</nomeUsual>
  <comentario>O êxito do Sushi depende de um bom Sumeshi</comentario>
  <ingrediente qtd="600" und="g" descr="Arroz de grão curto (nishiki)"> </ingrediente>
  <ingrediente qtd="750" und="ml" descr="Água"> </ingrediente>
  <ingrediente qtd="125" und="ml" descr="Vinagre de arroz"> </ingrediente>
  <ingrediente qtd="55" und="g" descr="Açúcar"> </ingrediente>
  <ingrediente qtd="0.5" und="colher de chá" descr="Sal"> </ingrediente>
  <processo>
    <duracao>Cerca de 1h15 de trabalho.</duracao>
    <passo>Lavar o arroz até que a água saia limpa e
      repousar num coador pelo menos 30 minutos.</passo>
    <passo>Os restantes 45 minutos são completo segredo!</passo>
  </processo>
</receita>
```

1. declaração XML

2. elemento documento ou elemento raiz

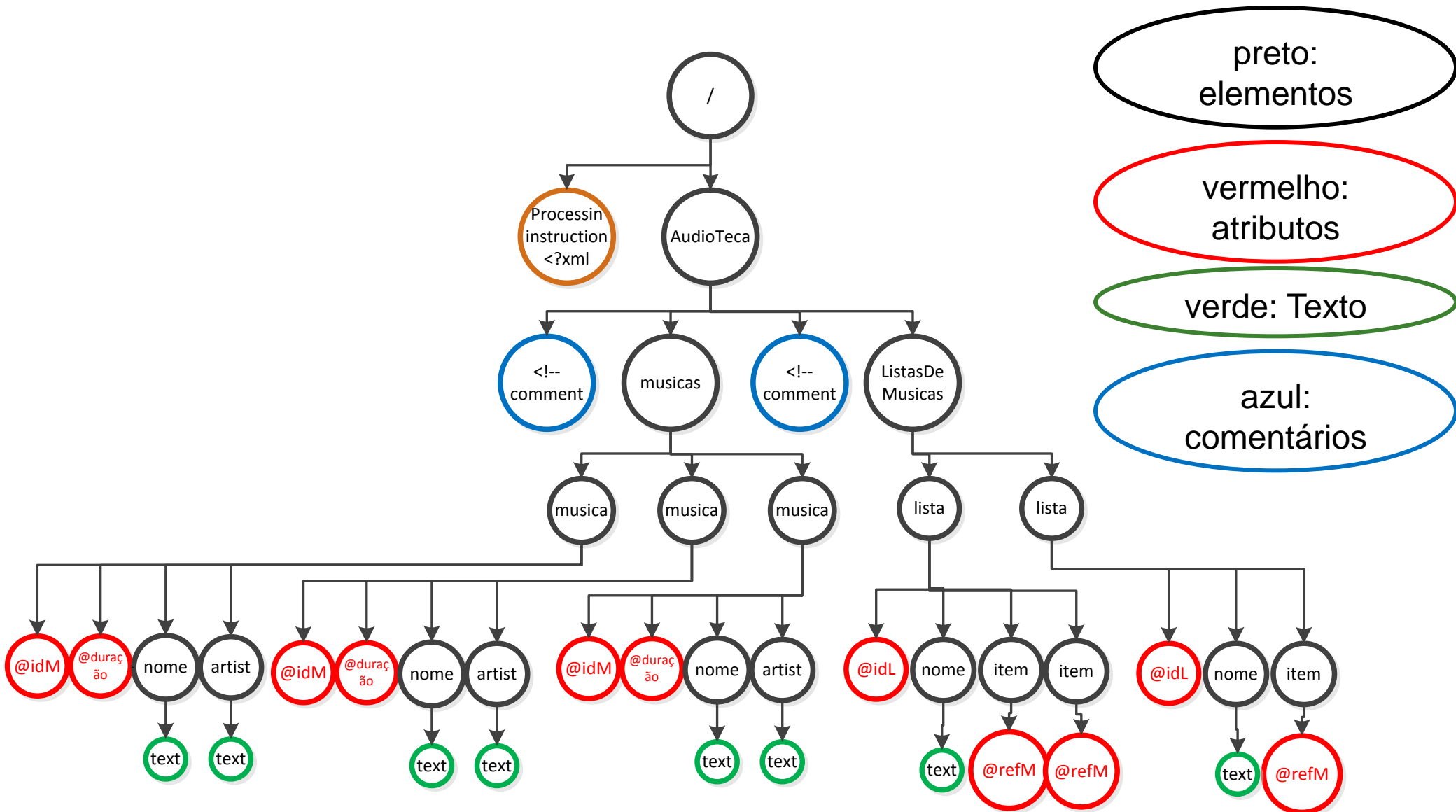
Representar informação de músicas e playlists

EXEMPLO

Representação em forma de árvore (1)

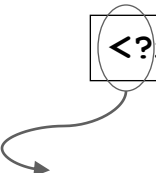
```
<?xml version="1.0" encoding="UTF-8"?>
<AudioTeca>
  <!-- Musicas existentes (a duração está em segundos) -->
  <musicas>
    <musica idM="m01" duração="150">
      <nome>Try!</nome>
      <artist>Pink</artist>
    </musica>
    <musica idM="m02" duração="100">
      <nome>Girl on Fire</nome>
      <artist>Alicia Keys</artist>
    </musica>
    <musica idM="m03" duração="206">
      <nome>Feel so close</nome>
      <artist>Calvin Harris</artist>
    </musica>
  </musicas>
  <!-- PlayLists-->
  <listasDeMusica>
    <lista idL="L01">
      <nome>Lista de relax</nome>
      <item refM="m02" />
      <item refM="m01" />
    </lista>
    <lista idL="L02">
      <nome>Lista de dança</nome>
      <item refM="m03" />
    </lista>
  </listasDeMusica>
</AudioTeca>
```

Representação em forma de árvore (2)



Declaração XML

- Um documento XML deve iniciar com uma declaração XML
 - ❑ embora isso não seja obrigatório
- Caso tenha declaração XML tem que estar no início do documento
 - ❑ não pode ter nada antes
 - ❑ ...nem espaços ou linhas em branco, nem comentários, ... nada!
- A declaração XML tem formato: `<?xml atributos?>`
 - ❑ onde os atributos são: `version`, `encoding` e `standalone`



```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

- ... passagem de informação ao analisador XML (“parser”)
 - ❑ ... idêntico à “instrução de processamento” (a ver posteriormente)
 - ❑ mas uma instrução de processamento escrever-se em qualquer local
 - ❑ a declaração XML, quando existe, está logo no início do documento!

Declaração XML: atributos

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

Obrigatório

Valor: "1.0" ou "1.1"

Opcional

Omissão: "UTF-8"

Opcional

Omissão: "yes"

Definição da mapa de codificação de caracteres a utilizar:

Latin-1 (tem caracteres acentuados): "ISO-8859-1"

Antiga codificação Windows do Latin-1: "windows-1252"

Indicação de existência de informação externa necessária à boa interpretação do documento:

Não existe informação externa: **standalone="yes"**

Existe informação externa (em ficheiro DTD): **standalone="no"**

(DTD ≡ "Document Type Definition", a detalhar posteriormente)

Caracteres acentuados (exemplo)

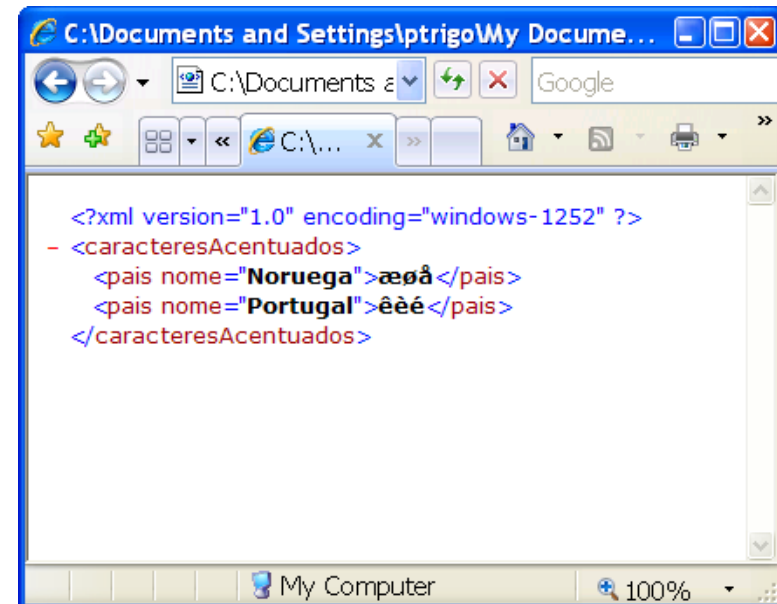
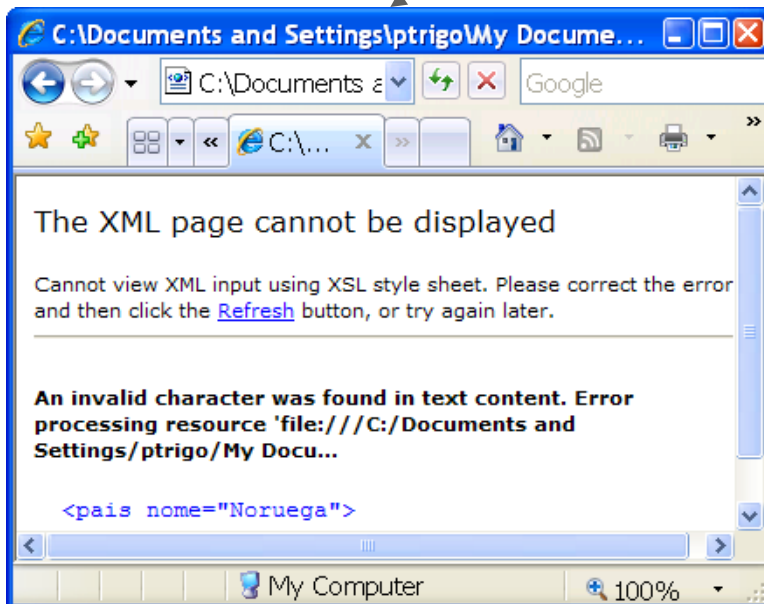
```
<?xml version="1.0"?>

<caracteresAcentuados>
  <pais nome="Noruega">æøå</pais>
  <pais nome="Portugal">êëé</pais>
</caracteresAcentuados>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<caracteresAcentuados>
  <pais nome="Noruega">æøå</pais>
  <pais nome="Portugal">êëé</pais>
</caracteresAcentuados>
```

O que devolve o analisador XML (“parser”)?



Aspectos gerais da linguagem XML

- Um documento tem sempre um único elemento raiz (“root element”)
 - todos os restantes elementos se definem no contexto do elemento raiz
 - e.g. `<receita> ... </receita>`
- As marcas (“tags”) são “case sensitive” e não podem ter espaços
 - e.g. `<nomeUsual>` \neq `<nomeusual>` e em `<nome usual>` temos erro
- Não se pode omitir a marca (“tag”) de terminação
 - e.g. bem formado em HTML e não em XML: `<p>Olá`
- O valor dos atributos é sempre escrito entre aspas ou plicas
 - e.g. `<ingrediente qtd="600"> </ingrediente>`
- Os elementos têm que estar adequadamente aninhados
 - e.g. bem formado em HTML e não em XML: `<i>Olá</i>`
 - ... em XML teria que ser: `<i>Olá</i>`

Aspectos gerais da linguagem XML (cont.)

- No XML os espaços em branco (no texto) são preservados
 - no HTML múltiplos espaços são reduzidos a um único
- No XML uma nova linha é sempre registada como LF (“line feed”)
 - o par CR / LF (“carriage return / line feed”) é convertido em LF
 - ... CR ≡ na máquina de escrever, empurrar o carroto até início da linha
 - ... LF ≡ na máquina de escrever, rodar carroto para nova linha
 - ... no Windows uma nova linha ≡ par CR / LF
 - ... no Linux uma nova linha ≡ LF (adoptado também pelo XML)
 - ... no Macintosh uma nova linha ≡ CR
- Um comentário XML escreve-se
 - `<!-- Isto é um comentário -->`
- Em síntese: o XML é simplesmente texto com marcas (“tags”)
 - as aplicações dão uma interpretação a essas marcas...

O documento XML e o seu analisador (“parser”)

- Um documento XML diz-se bem formado
 - ❑ se respeitar as regras da linguagem XML
 - ❑ e.g. “o valor dos atributos é delimitado por aspas ou plicas”, etc
- Quem diz se um documento XML está bem formado?
 - ❑ um analisador (“parser”) XML – aplicação que verifica que um ficheiro XML está bem construído
 - ❑ e.g. o “browser”, o “Notepad XML”, o “XML Spy”, ... (estes programas têm um analisador XML interno)
- A especificação W3C do XML diz que um analisador XML
 - ❑ deve interromper a análise assim que detectar um erro
 - ❑ ... e considerar que o documento não está bem formado
- ... um documento HTML pode ter erros (e.g. esquecer fechar marca)
 - ❑ o que contribui para se ter “browsers” “grandes e “incompatíveis”!
 - ❑ ... se há erro, cada um tem modo próprio de apresentar o documento!

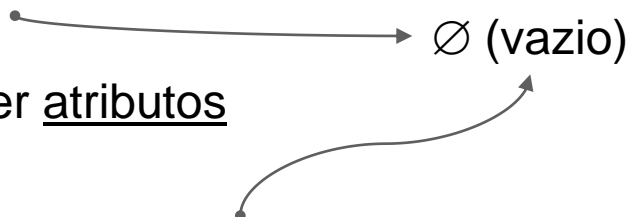
Em síntese: o que é um elemento?

- Um elemento é tudo desde (incluindo) a
 - marca de início do elemento até à marca de fim (incluindo) de elemento
 - e.g. o elemento `receita`
- O elemento relaciona-se com os restantes
 - de acordo com a sua posição na hierarquia de elementos
 - e.g. o elemento `nomeOriginal` é irmão de `ingrediente`

```
<receita>
  <nomeOriginal>Sumeshi</nomeOriginal>
  <comentario>O êxito do Sushi depende de um bom Sumeshi</comentario>
  <ingrediente qtd="600" und="g" descr="Arroz de grão curto (nishiki)"> </ingrediente>
  <ingrediente qtd="750" und="ml" descr="Água"> </ingrediente>
</receita>
```

Que elementos aqui estão? `receita`, `nomeOriginal`, `comentario`, `ingrediente`, `ingrediente`
dois elementos diferentes!

O elemento e o seu conteúdo

- O conteúdo de um elemento é tudo o que está entre
 - as marcas (exclusive) de início e de fim de elemento
- O conteúdo de um elemento pode ser
 - simples, se apenas contiver texto
 - complexo, se contiver outros elementos
 - vazio, se não contiver qualquer informação
 - ... qualquer dos anteriores pode sempre conter atributos
- Qual o conteúdo deste elemento?
`<ingrediente qtd="750" und="ml" descr="Água"></ingrediente>`

Ø (vazio)
- Um elemento de conteúdo vazio ou simplesmente elemento vazio
 - pode escrever-se iniciando com < e terminando apenas com />`<ingrediente qtd="750" und="ml" descr="Água"/>` (bem formado)

O elemento e o seu nome

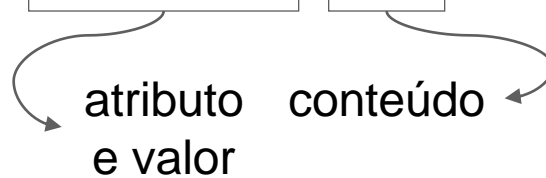
- O nome de um elemento é um xml Name
 - ❑ `Name ::= NameStartChar (NameChar)*`
 - ❑ `NameStartChar ::= ":" | [A-Z] | "_" | [a-z]`
 - ❑ `NameChar ::= NameStartChar | "-" | "." | [0-9]`
- Evitar usar os caracteres
 - ❑ sinal de subtracção (-) e ponto (.) para reduzir possível ambiguidade
 - ❑ dois pontos (:) pois é reservado para utilização em “namespaces”
- Numa organização é boa prática adoptar as convenções
 - ❑ já existentes para a nomenclatura das bases de dados da organização

Indicação do tipo “referência a entidade”

- O conteúdo de um elemento não pode conter o carácter <
 - pois será interpretado como início de marca
 - exemplos:
 - `<info>Uma tag delimita-se por < e ></info>` (ERRO)
 - `<info>Uma tag delimita-se por < e ></info>` (OK)
- O `&x` designa-se por referência a entidade (“entity reference”)
 - diz ao “parser” que substitua `&x` pelo texto representado por `x`
- O XML predefine 5 referências a entidade
 - `<` para `<`, `&` para `&`, `>` para `>`, `"` para `"`, `&apos` para `'`
- Apenas `<` e `&` são obrigatórias no conteúdo de um elemento
 - as restantes são úteis, por exemplo, em valores de atributos
 - `<receita nome="Arroz de "Sushi""></receita>`

O elemento e os seus atributos

- Um elemento pode ter atributos definidos no início de uma marca
 - ❑ `<ingrediente>Água</ingrediente>` (sem atributos)
 - ❑ `<ingrediente qtd="750">Água</ingrediente>` (com atributo)



- O valor do atributo é sempre escrito entre aspas ou plicas
 - ❑ se o valor contém plicas tem que ser escrito entre aspas e vice versa
 - ❑ e.g. `<receita nome='Arroz de "Sushi"'></receita>`
 - ❑ e.g. `<receita nome="Arroz de 'Sushi'"></receita>`
 - ❑ ... em geral, na ausência de outra imposição, escreve-se entre aspas
 - ❑ ... e para os casos complicados lá estão `"` (") e `&apos` (')

Utilizar elemento ou atributo?

■ Que modelo escolher?

❑ apenas elementos?

```
<receita>
  <nomeOriginal>Sumeshi</nomeOriginal>
  <nomeUsual>Arroz de Sushi</nomeUsual>
</receita>
```

❑ apenas atributos?

```
<receita nomeOriginal="Sumeshi" nomeUsual="Arroz de Sushi">
</receita>
```

❑ elementos e atributos?

```
<receita nomeOriginal="Sumeshi">
  <nomeUsual>Arroz de Sushi</nomeUsual>
</receita>
```

... elemento ‘versus’ atributo: características

	Elemento	Atributo
Pode conter múltiplos valores?	Sim	Não
Pode representar estruturas?	Sim	Não
É mais simples de expandir (alterações futuras)?	Sim	Não
É mais simples de manipular via código?	Sim	Não

- Então porquê pensar em elemento ‘versus’ atributo?
 - talvez porque “as soluções mais simples são as melhores!
 - ... desde que não sejam simples demais” (e quem disse isto?...)
- Recorde-se o modelo relacional
 - só com atributos (e domínio atómico) é referência incontornável!

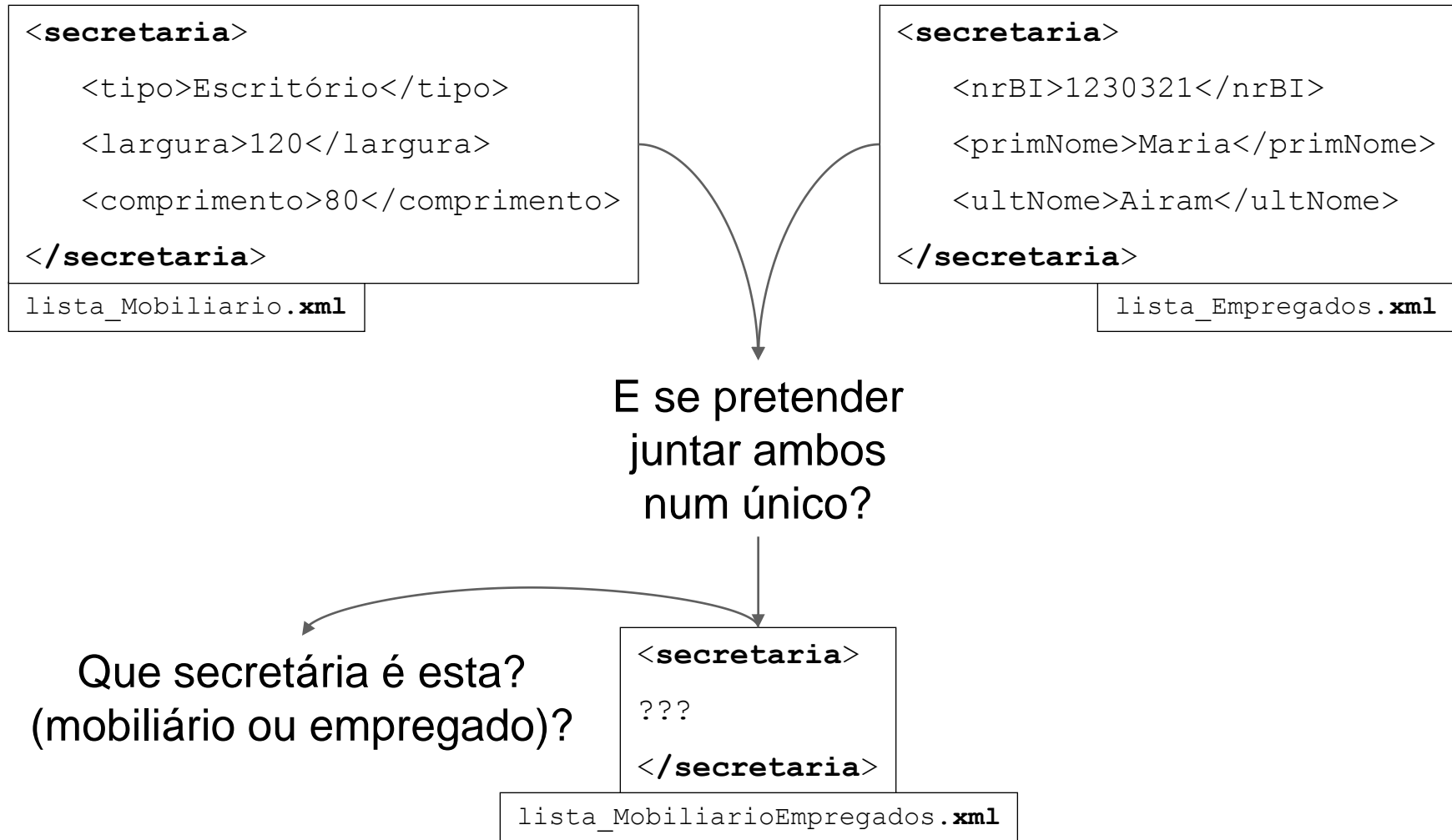
... elemento 'versus' atributo: discussão

- Há quem defenda a seguinte abordagem
 - atributo ≡ para descrever algo sobre dados (meta-dados)
 - elemento ≡ para, no seu conteúdo, representar os dados

```
<biography xmlns:xlink="http://www.w3.org/1999/xlink/namespace/">
  <image source="http://www.turing.org.uk/turing/scrapbook/run.html"
    width="152" height="345"/>
  <person born="1912-06-23" died="1954-06-07">
    <first_name>Alan</first_name>
    <last_name>Turing</last_name></person>
  was one of the first people to truly deserve the name
  <emphasize>computer scientist</emphasize>.
</biography>
```

- Há quem defenda que: nem sempre se consegue perceber
 - para determinado problema, o que são dados e o que são meta-dados!
 - ... tudo depende da utilização que se pretende dar aos dados

Conflitos entre nomes



Resolução de conflitos: espaço de nomes e nome qualificado



Espaço de nomes (“namespace”)

- Para que serve?
 - ❑ para distinguir elementos e atributos de diferentes vocabulários
 - ❑ ... portanto com diferentes significados, mas com o mesmo nome,
 - ❑ para agrupar elementos e atributos que se relacionam numa aplicação.
- Implementa-se admitindo que
 - ❑ elementos e atributos com o mesmo URI têm o mesmo “namespace”

```
<emp:secretaria xmlns:emp="http://Empregados">  
  <nrBI>1230321</nrBI>  
  <primNome>Maria</primNome>  
  <ultNome>Airam</ultNome>  
</emp:secretaria>
```

URI que representa
o espaço de nomes
("namespace")

URI, URL e URN

- Um “Uniform Resource Identifier” (URI) é
 - ❑ cadeia de caracteres que identifica um recurso na Internet, na forma,
 - ❑ `<schemeName> : <hierarchicalPart> [?<query>] [#<fragment>]`
 - ❑ e.g. `asReceitas:c/exemplo/aqui?nome=receita#sushi`
 - ❑ `schemeName` comuns: `http`, `https`, `file`, `nfs`, `telnet`, `mailto`, etc
- O URI mais conhecido é o “Uniform Resource Locator” (URL)
 - ❑ que especifica como aceder ao recurso (e.g. a localização numa rede)
 - ❑ e.g. `http://exemplo.org:8080/aqui?nome=receita#sushi`
 - ❑ ... indica protocolo, e adere às convenções de nomes na Internet
 - ❑ e.g. `ftp://user.password@host.port/path`
- Outro URI conhecido é o “Uniform Resource Name” (URN)
 - ❑ identifica recurso mas não implica a sua localização(o URL implica)
 - ❑ tem formato: `urn:namespaceID:namespaceSpecificString`
 - ❑ ... URN costuma comparar-se ao ISBN (e.g. `urn:isbn:978-0321122261`)
 - ❑ ... o URN permite “saber qual é o livro”; o URL permite “obter o livro”

O atributo `xmlns` (XML “namespace”)

- Os conflitos entre nomes eliminam-se adicionando
 - um prefixo a cada nome de elemento ou de atributo
- O atributo `xmlns` usa-se para associar um `prefixo` a um URI
 - e tem o formato: `xmlns:prefixo="URI"`
 - e.g. `<...xmlns:oMeuPrefixo="http://oMeuEspacoDeNomes"`
- Então porque não utilizar o URI como prefixo?
 - o URI pode conter caracteres, e.g. `/`, `%` e `~`, não válidos em nomes XML
- Qualquer nome `nome` prefixado com `prefixo` fica com a forma
 - `prefixo:nome`
 - e.g. `emp:secretaria` refere um nome no espaço atrás indicado

Terminologia – nome qualificado, prefixo e parte local

“qualified name” ou “QName” ou “raw name”
nome qualificado

`oMeuPrefixo:receita`

“prefix”
prefixo

“local part”
parte local

previamente
associado a um URI

um nome que fica
associado ao “namespace”

`xmlns:oMeuPrefixo=`
`“http://oMeuEspacoDeNomes”`

Que nomes pertencem a um “namespace”?

- Quando um “namespace” é definido num elemento (atributo `xmlns`)
 - ❑ o contexto da definição é o desse elemento e do seu conteúdo
 - ❑ qualquer descendente com esse prefixo está no mesmo “namespace”
 - ❑ ... sem prefixo, está no “namespace” de omissão, se este foi definido

```
<emp:secretaria xmlns:emp="http://Empregados">
  <emp:nrBI>1230321</emp:nrBI>
  <nomeCompleto>
    <emp:primNome>Maria</emp:primNome>
    <ultNome>Airam</ultNome>
  </nomeCompleto>
</emp:secretaria>
```

... este nome está no
“namespace”
`http://Empregados`?

	Sim	Não
secretaria	√	
nrBI	√	
nomeCompleto		√
primNome	√	
ultNome		√

nomes locais: não estão em nenhum “namespace”

O “namespace” de omissão

- Sabendo que determinados nomes estão no mesmo “namespace”
 - ❑ pode definir-se um “namespace” de omissão
 - ❑ com o formato: `xmlns="URI"`

```
<emp:secretaria xmlns:emp="http://Empregados"
  xmlns="http://EmpOmissao">
  <emp:nrBI>1230321</emp:nrBI>
  <nomeCompleto>
    <emp:primNome>Maria</emp:primNome>
    <ultNome>Airam</ultNome>
  </nomeCompleto>
</emp:secretaria>
```

... este nome está no
“namespace”
`http://Empregados?`

	Sim	Não
secretaria	√	
nrBI	√	
nomeCompleto		√
primNome	√	
ultNome		√

estão no “namespace” `http://EmpOmissao`

O “namespace” de omissão: atributos

- No contexto da definição de um “namespace” de omissão
 - os elementos não qualificados (sem prefixo) estão nesse “namespace”
 - ... mas o “namespace” de omissão apenas se aplica a elementos
 - ... os atributos não qualificados não pertencem a nenhum “namespace”

está em <http://www.w3.org/2000/svg?>
 (“Scalable Vector Graphics”)

```
<svg xmlns="http://www.w3.org/2000/svg"
  width="12cm" height="10cm">
  <ellipse rx="110" ry="130" />
  <rect x="4cm" y="1cm"
    width="3cm" height="6cm"/>
</svg>
```

	Sim	Não
svg	√	
width		√
height		√
ellipse	√	
rx		√
ry		√
rect	√	
x		√
y		√

nomes locais: não estão em nenhum “namespace”

Qual o processamento sobre um “namespace”?

- Em geral, o URI que identifica um “namespace”
 - ❑ apenas fornece um nome único a esse “namespace”
 - ❑ não é usado pelos analisadores (“parsers”) XML para aceder ao recurso
- Os analisadores XML verificam essencialmente
 - ❑ se todos os prefixos foram previamente associados a um URI
 - ❑ ... excepto os prefixos `xmlns` e `xml` (essa associação está predefinida)
- Outras aplicações que trabalhem sobre um analisador XML
 - ❑ podem tratar os elementos dependendo do seu “namespace”
 - ❑ e.g. analisador XSD (“XML Schema Definition”)
 - ❑ e.g. motor XSLT (“eXtensible Stylesheet Language Transformations”)

Sobre o “parser”: espaços de nomes predefinidos

- Um analisador (“parser”) XML têm informação ‘a priori’
 - sobre alguns espaços de nomes, e.g. `xml` e `xmlns`
- `xml` é um espaço de nomes predefinido
 - está associado a `http://www.w3.org/XML/1998/namespace`
 - não precisa ser declarado e não pode ser redefinido
- `xmlns` é um espaço de nomes predefinido
 - está associado a `http://www.w3.org/2000/xmlns/`
 - não pode ser declarado nem redefinido
 - nele não podem ser definidos outros nomes
 - apenas se utiliza para definir associações a espaços de nomes

```
<descricao xml:space="preserve">  
Este texto mantém os espaços em branco.  
</descricao>
```

e.g. indicação de
que as aplicações
devem preservar os
espaços em branco

O “parser” analisa todo o texto de um documento XML?

- Não. “Existem pequenas ilhas que sobrevivem”...
 - tudo o que estiver escrito dentro de uma secção `CDATA` é ignorado!
- A secção `CDATA` (“Character Data”) define-se
 - `<![CDATA[Este texto não é analisado: < e estão Ok]]>`
 - e.g. usar para conter texto com grande quantidade de `<` e `&`
 - e.g. o código de uma função

```
<umScript>
  <![CDATA[
    function between(a,b1,b2) {
      if (b1 < a && a < b2) then return 1
      else return 0
    ]]>
</umScript>
```

uma secção `CDATA`
não pode conter `]]>`
↓
`CDATA` não pode conter
outras secções `CDATA`

Como passar, no XML, informação às aplicações?

- Usando instruções de processamento (“processing instructions”)
 - ❑ com o formato: `<?target processingInstruction?>`
 - ❑ target é um qualquer nome com sentido para determinada aplicação

```
<?robots index="yes" follow="yes"?>
```

Diz aos motores de busca para indexar página e seguir “links”

```
<?php
```

```
mysql_connect("db", "eu", "senha");  
$result = mysql("HR", "SELECT * FROM T ORDER BY A1");  
$i = 0;  
while($i < mysql_numrows($result)){  
    $fields = mysql_fetch_row($result);  
    echo "<person>$fields[1] $fields[0] </person>\r\n";  
    $i++;}  
mysql_close();
```

```
?>
```

Diz ao interpretador PHP
("PHP: Hypertext Preprocessor")
para gerar elementos person a
partir de uma base de dados

O que o XML fornece e o que não fornece...

- O XML permite especificar
 - ❑ a estrutura de um documento através de uma hierarquia de marcas
 - ❑ diferentes espaços de nomes para eliminar ambiguidade
 - ❑ passagem de informação específica de determinados analisadores
- O XML não permite especificar
 - ❑ as marcas que são válidas para determinado domínio
 - ❑ uma estrutura de marcas para determinado domínio
 - ❑ uma forma de navegação na estrutura de marcas
 - ❑ a forma de apresentar o documento

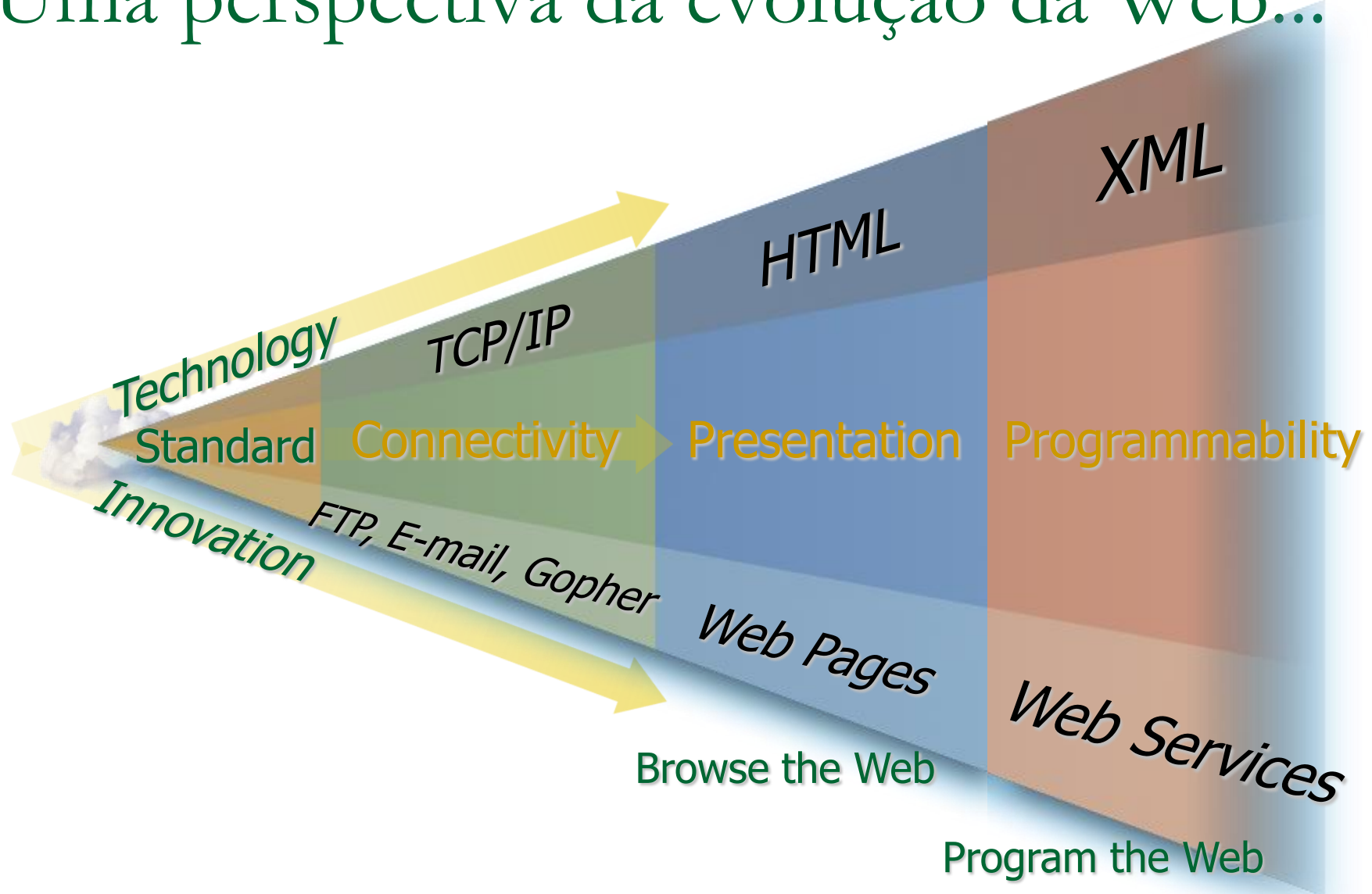
Qual o próximo passo?

- Especificar as estruturas de marcas que um documento deve seguir
 - ... favorecendo assim o uso do XML para intercâmbio de informação
 - recorrendo às linguagens:
 - DTD (“Document Type Definition”), e/ou
 - XSD (“XML Schema Definition”)
- Definir formas de navegar na estrutura de marcas de um documento
 - usando a linguagem XPath (“XML Path Language”)
- Definir formas de apresentar um documento, recorrendo ao
 - XSL (“eXtensible Stylesheet Language”) e suas componentes:
 - XSLT (“XSL Transformations”),
 - XPath (“XML Path Language”), e
 - XSL-FO (“XSL Formatting Objects”).

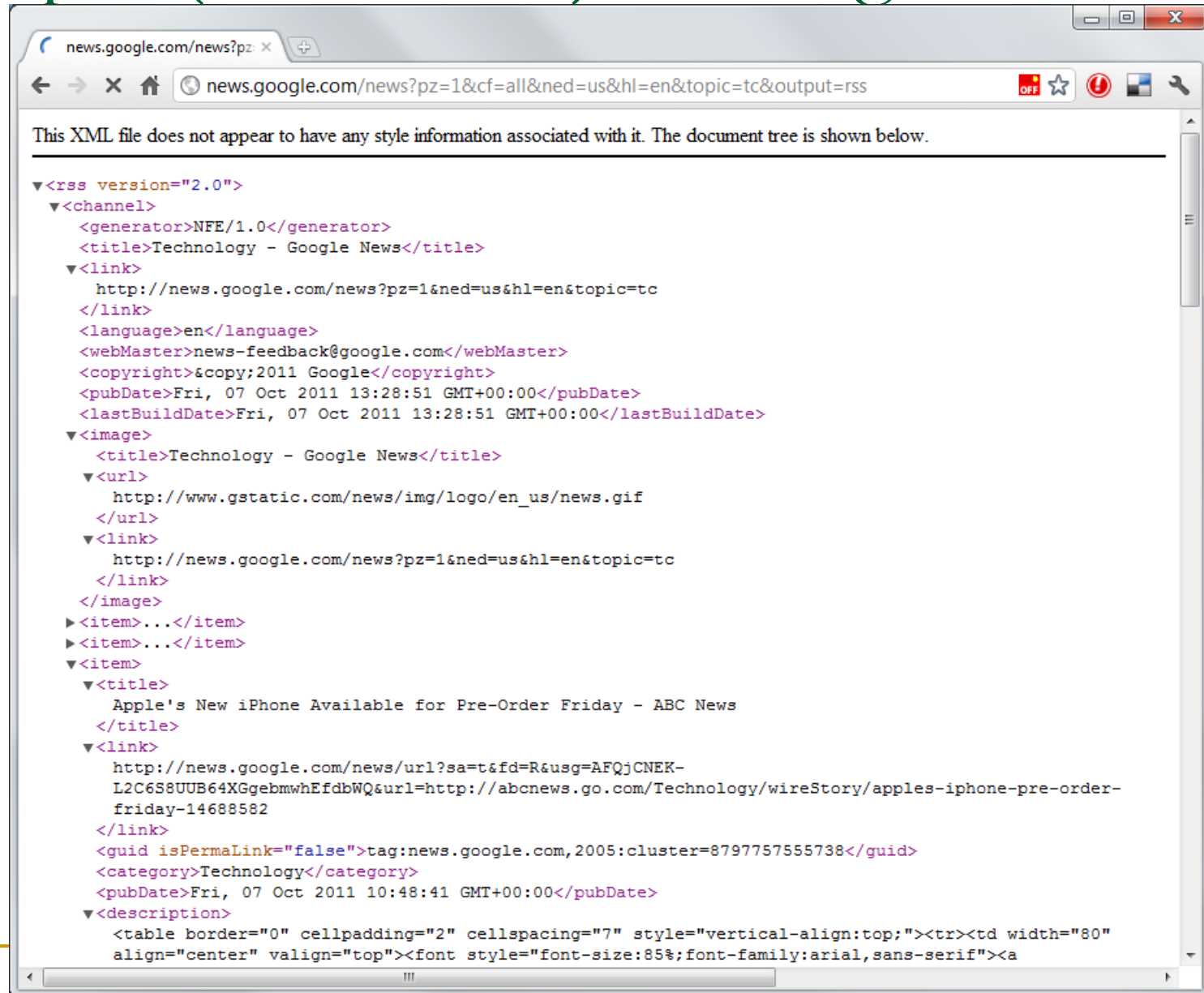
... a actual utilização típica do XML

- Transferência de dados
 - formato de texto
 - independente do software e do hardware
- Assumindo acordo quanto às marcas usadas e sua estrutura
 - pode ser lido por qualquer aplicação que adopte esse acordo
 - ... acordo pode ser imposto por DTD e/ou XSD
- Simplifica integração de sistemas distintos
 - por admitir um formato único para representar as estruturas de dados

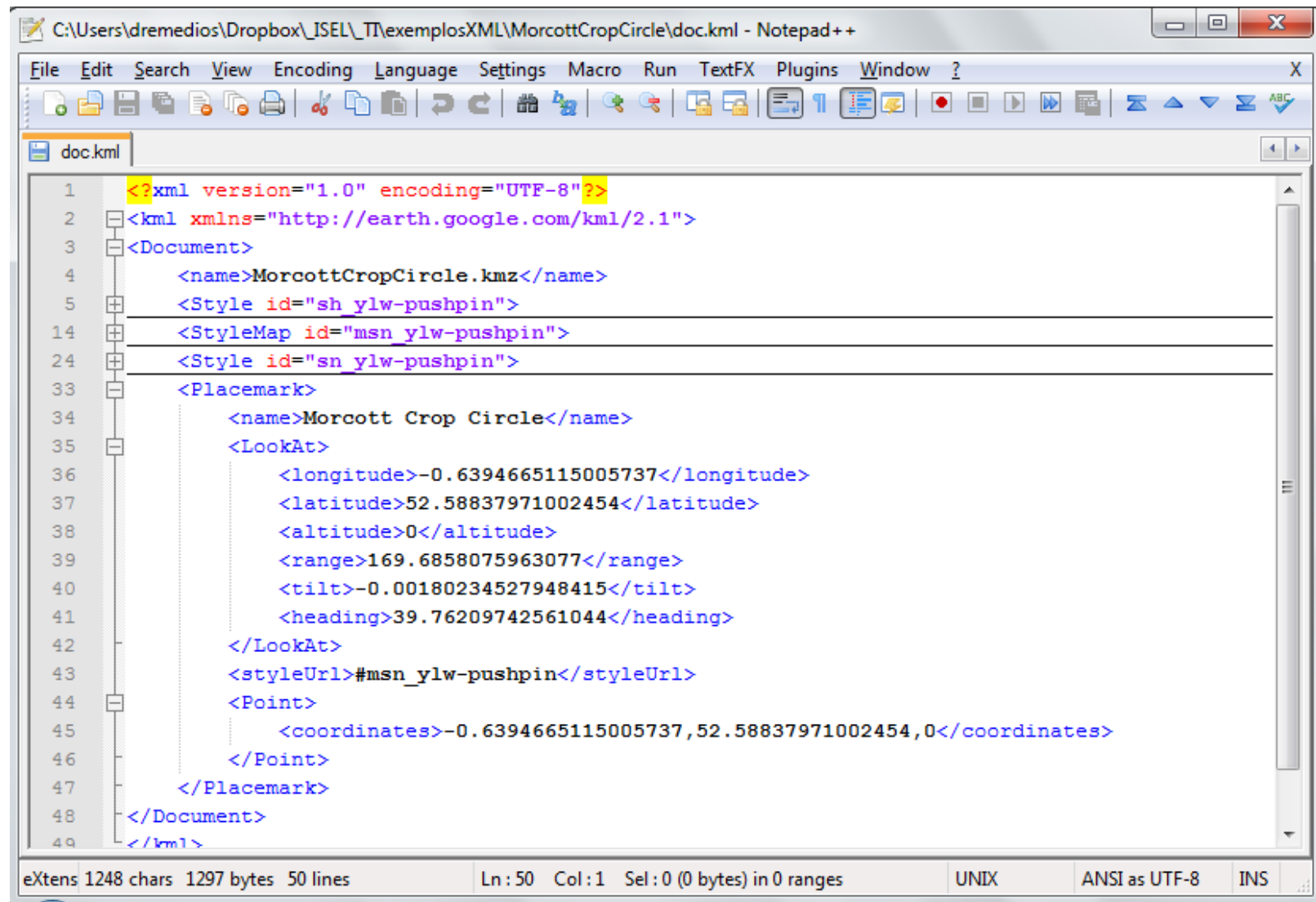
Uma perspectiva da evolução da Web...



Exemplo (RSS Feeds) – Google News



Google PlaceMark – kmz (zip of kml files)



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kml xmlns="http://earth.google.com/kml/2.1">
3  <Document>
4      <name>MorcottCropCircle.kmz</name>
5      <Style id="sh_ylw-pushpin">
14     <StyleMap id="msn_ylw-pushpin">
24     <Style id="sn_ylw-pushpin">
33     <Placemark>
34         <name>Morcott Crop Circle</name>
35         <LookAt>
36             <longitude>-0.6394665115005737</longitude>
37             <latitude>52.58837971002454</latitude>
38             <altitude>0</altitude>
39             <range>169.6858075963077</range>
40             <tilt>-0.00180234527948415</tilt>
41             <heading>39.76209742561044</heading>
42         </LookAt>
43         <styleUrl>#msn_ylw-pushpin</styleUrl>
44         <Point>
45             <coordinates>-0.6394665115005737,52.58837971002454,0</coordinates>
46         </Point>
47     </Placemark>
48 </Document>
49 </kml>
```

eXtens 1248 chars 1297 bytes 50 lines Ln: 50 Col: 1 Sel: 0 (0 bytes) in 0 ranges UNIX ANSI as UTF-8 INS

SVG – Scalable Vector Graphics

```
<html>
<body>

<p><b>Note:</b> This example only works in Firefox and Google
Chrome.</p>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect x="20" y="20" width="250" height="250"
  style="fill:blue">
    <animate attributeType="CSS" attributeName="opacity"
    from="1" to="0" dur="5s" repeatCount="indefinite" />
  </rect>
</svg>

</body>
</html>
```

Note: This example only works in Firefox and Google Chrome.



```
<html>
<body>

<p><b>Note:</b> This example only works in Firefox and Google
Chrome.</p>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <g transform="translate(100,100)">
    <text id="TextElement" x="0" y="0" style="font-
    family:Verdana;font-size:24"> It's SVG! Exemplo de TI
    <animateMotion path="M 0 0 L 100 100" dur="5s"
    fill="freeze" />
    </text>
  </g>
</svg>

</body>
</html>
```

Note: This example only works in Firefox and Google Chrome.

It's SVG! Exemplo de TI